Design and implementation of hybrid vehicle using control of DC electric motor

Muhammad Majid Hussain
Al Rashid Mustafa
Muhammad Akmal Chaudary
Abdul Razaq

The final published version is available from doi:
https://doi.org/10.1109/UPEC.2019.8893604

Design and Implementation of Hybrid Vehicle using Control of DC Electric Motor

Muhammad Majid Hussain
Faculty of Computing, Engineering and Science
University of South Wales
Cardiff, United Kingdom
muhammad.hussain@southwales.ac.uk

Al Rashid Mustafa
The Centre for Automotive & Power Systems Engineering
University of South Wales
Cardiff, United Kingdom
rashid.mustafa@southwales.ac.uk

Muhammad Akmal Chaudary
Department of Electrical and Computer Engineering
Ajman University
Ajman, United Arab Emirates
m.akmal@ajman.ac.ae

Abdul Razaq
School of Design and Informatics
Abertay University
Dundee, United Kingdom
a.razaq@abertay.ac.uk

***Abstract*—The electric motors and its control technology are key components of hybrid electric vehicles (HEVs). Control of the electric motor is a fundamental issue for traction application in electric vehicles and HEVs. This paper presents the design, development and implementation of a hybrid vehicle using both an electric motor and petrol engine to increase efficiency and decrease carbon footprint. Initially, a prototype of a HEV is designed and the performance values are calculated, before a control system is developed and implemented to control the DC motor speed using a microcontroller as the vehicle's electronic control unit along with simple proportional integral derivative (PID) control using speed as a feedback mechanism. The prototype made incorporated voltage, current, speed and torque sensors for feedback resulting in a closed loop control system which successfully matched the speed input of a user-controlled pedal sensor. A user interface was developed to show the driver of the vehicle key variables such as the revolutions per minute (RPM) of the motor, the speed of the vehicle along with the current being drawn, and the voltage applied to the motor with overall power. To output a variable voltage from the Arduino, a digital output was used with pulse width modulation (PWM) capabilities in order to provide a variable DC voltage to the speed controller.**

Keywords— DC motor, hybrid electrical vehicles, PID controller, speed control, user interface

I. INTRODUCTION

The model of a HEV is multifaceted with many different components. Each component needs to be modelled. The design of each component is a difficult task as the parameters of one component can devastate the power level of others. The consequences of inappropriate power might make the vehicle unnecessarily expensive or inefficient [1]. In an urban area, as a result of emission-free and environmentally friendly zones, HEVs represent a key factor in improvement of traffic and moreover reduction of the carbon footprint. The major requirement for traction motors used in HEVs is to generate propulsion torques over an extensive speed range. Two main types of motor are commonly used for HEVs – the permanent magnet motor (PMM) and the induction motor (IM).

Problems relating to the energy crisis and extensive carbon emission have become more and more serious, and electric vehicles (EVs) seem to be the ideal solution. As batteries have too low energy density and charging and discharging issues, the alternative choice is the HEV. Different types of DC motors have been used in HEVs, including the brushless DC motor, which has the benefits of light weight and high efficiency [2, 3]. An electric motor is one of the HEV's core components. The important aspects of electric motors used in HEVs are high power density, high starting torque, high efficiency, robustness, good reliability and wide range of speed control [4, 5].

The use of electric motors in everyday life has become very significant. Generally, electric motors are used in conveyors, air-conditioning and EVs. Technically, the HEV contains four main parts for its drive: the control system and motor and its driving system, as well as energy storage and its body, of which the motor and its drive system determine the overall attributes of EVs. When designing EVs, we should consider the main performance requirements – for example, quick start and brake, variable speed, large load and high power. Numerous methods are already used in the speed control of DC motors, one of which is PID [6]. PID has a simple structure and has advantages in each framework. The application of a PID controller for a HEV is presented in the literature [7, 8].

The main aim of this paper is to design and develop a hybrid vehicle prototype, simulate it, and analyse the effect of incorporated voltage, current, speed and torque. In addition, a new prototype to control DC motor speed using a PID microcontroller of HEV is developed. The proposed HEV prototype is assembled with a simple design procedure and low-cost available equipment. The design of the PID controller is subsequently introduced to tune the PID controller's three parameters, K_p, K_i and K_d, based on proving the functionality of the proposed algorithm from a set of simulations.

II. TYPES OF HYBRID TECHNOLOGIES

Two main types of hybrid technology are used today – parallel hybrid and series hybrid vehicles [9]. The differences come in the drivetrain technology and how the power is delivered to the wheels. There are advantages and disadvantages in both cases and deciding on a particular type of hybrid car heavily depends on what kind of journeys the car is going to be making or what the car is being used for. Figures 1 and 2 show the types of hybrid vehicle technologies.

978-1-7281-3349-2/19/$31.00 ©2019 IEEE

A. *Series Drivetrain*

In this, the electric motor provides power to the wheels directly. Typically, a small electric engine either charges the batteries as required or more rarely is used to power the electric motor directly. This is one of the simplest forms of hybrid vehicle. Series hybrid vehicles are much more efficient in start–stop traffic, benefitting from instant torque at low speeds. However, the larger battery pack and the need for a generator can often make it more expensive.

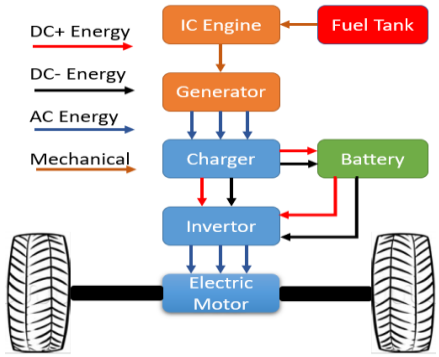

Fig. 1. Series hybrid system.

B. *Parallel Drivetrain*

In this, both the petrol engine and the electric motors are used to provide power to the wheels in a parallel system which can provide performance increases as well as increased efficiency. As a result of the regenerative braking or coasting system in parallel cars which utilise the engine's energy in moments of low demand, the battery pack can often be smaller. This system is better for higher speeds and longer ranges as it is less reliant on the electric motor. It still, however, benefits in start–stop conditions, simultaneously using the electric motor to boost efficiency and performance.

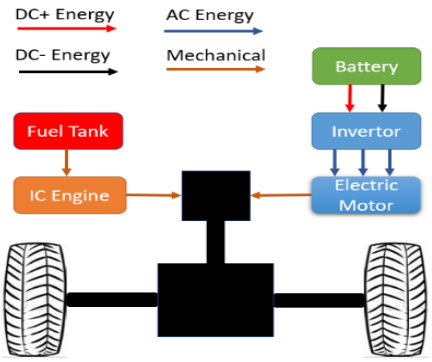

Fig. 2. Parallel hybrid system.

III. Design and Development

A flow chart has been developed for the MATLAB /Simulink software programme so far. This can be seen in figure 3, while figure 4 is a schematic diagram for prototype testing. The system starts by checking the initial system parameters such as motor voltage, battery voltage and current going to the motor. If all of these parameters are within range, ensuring the motor is safe to run, the liquid crystal display (LCD) is initialised and updated. The ignition switch is then checked along with the direction switches. If the system is ready to run, the PID control is implemented using the pedal sensor as the input (set point) and the PWM signal as the output using the speed of the shaft as the feedback. Once the PID control is implemented, the software code goes back, reading the current and voltages before updating the LCD screen. Once the LCD screen is updated, the hardware switches are integrated and the PID control implemented again. This worked well to prove the concept – however, the software code is optimised further when the LCD screen is successfully run off a separate microcontroller.

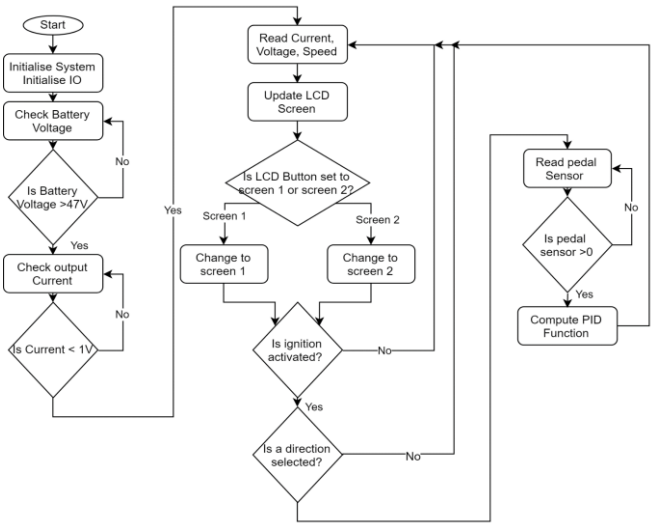

Fig. 3. Flow chart for proyotype design concept.

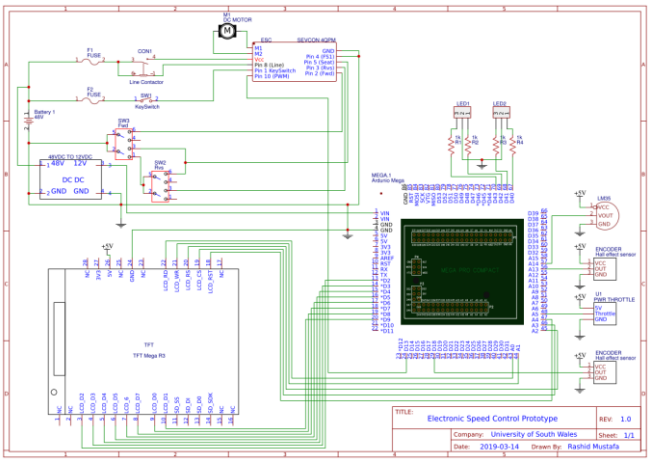

Fig. 4. Schematics of current electrical and electronic PID controlled motor prototype.

IV. Initial Testing of DC Motor

The motor obtained was manufactured by Lynch Motors in Honiton, UK, and is a LEM 170, which is a 48V, 5.54kW motor rated at 3264 RPM and 16.2Nm of torque. The motor was tested with a bench power supply to establish if it was operational. The no-load current of the motor was found to be around 6A and the motor was run with a varying supply voltage of up to 30VDC. It was recommended in the installation manual to let the motor run for half an hour with less than 30% full current if the motor had not been used in more than two years. During the test of the motor, it was therefore left running for half an hour supervised. The motor ran at variable speeds and worked as expected. Figure 5 shows the testing of the DC motor. It was found that, after the motor accelerated to the final speed allowed by the voltage, the current drawn was between 6-7 amps.

During the testing, the voltage was set at different levels and the speed of the shaft was measured with a digital tachometer by applying a reflective strip to the shaft. The

results from the test can be seen in figure 6, which shows a graph of the speed of the motor versus the voltage applied to the motor. It was found that the speed rose proportionally to the voltage applied, which was expected of this DC motor, and the final speed was found to be 3320 RPM, faster than stated in the datasheet. This is probably because no load was present on the motor.

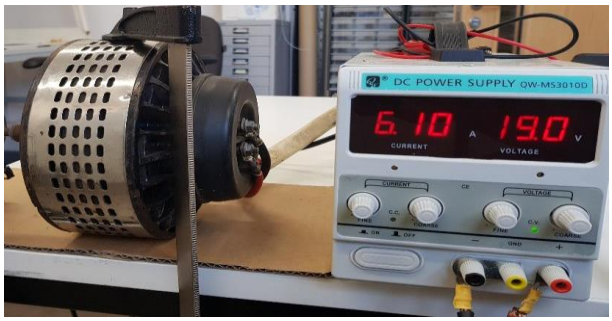

Fig. 5. Testing of the DC motor using a bench power supply.

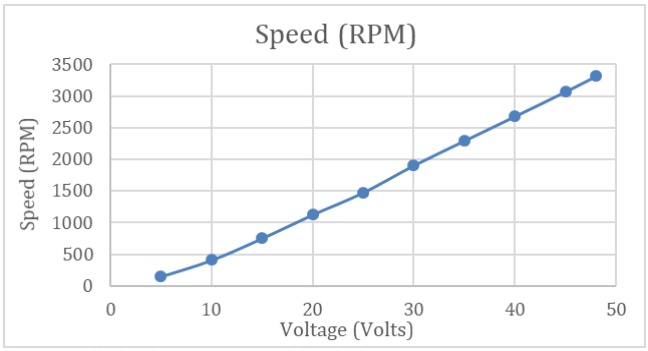

Fig. 6. Applied voltage vs RPM of the shaft.

V. Testing of Brushed DC Controller

A 400A brushed DC controller with high power protection and digital and analogue IO for control of the motor in different applications was designed and tested. This controller can be controlled from a potential divider, hall sensor or 0-5V variable power supply. The controller is pictured in figure 7. It was set up with the power supply and a prototype throttle as an input for initial testing. This was to confirm that the motor and controller worked together before adding any extra hardware and software. The setup can be seen in figure 8, and a simplified schematic of the connections made is set out in figure 9.

Fig. 7. 4PQM controller.

Fig. 8. 4QPM controller and LEM 170 DC Motor.

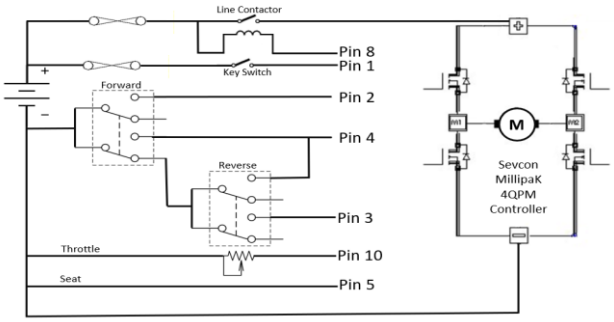

Fig. 9. Simplified schematics of wiring.

VI. Throttle Development for Electric Motor

The first step in introducing a microcontrolled input for the brushless DC motor controller was to procure a foot pedal and test read its output with a microcontroller. The microcontroller selected was an Arduino Nano and the foot pedal chosen was a standard foot pedal used on electric motorcycles. The device runs off a 5V input and gives an output signal of 1V to 4.2V depending on how much the pedal is pressed. A simple Arduino Nano was used to help with identification of the IO pins and communication protocols. The foot pedal was connected to a 5V supply and the output was connected to a multimeter to ensure that the output voltage was in the range of the Arduino's analogue reading capabilities. The Arduino can read up to 5V DC and, when testing the foot pedal, it was found that the output ranged from 0.9V to 4.3V, which is slightly different from the product description. There is no datasheet with this product, so it was manually tested. However, the output was found to be in the range of the Arduino's analogue reading capabilities.

The foot pedal was connected to the Arduino using one of the digital inputs and an analogue read function was used to read the voltage output of the pedal sensor. The Arduino uses a 10-bit analogue to digital converter (ADC) in the analogue ports. Using equation 1, it can be determined that the maximum range (5V) of the analogue input will be converted to 1024 divisions, otherwise known as ticks or divisions:

$$2^n = \text{Maximum deviation} \quad (1)$$

where n = the number of bits of the ADC

From this, the number of ticks for the minimum and maximum outputs of the pedal sensor can be calculated and inputted into the Arduino programme. If 5V is equal to 1024 ticks, then 0.9V represents 184 ticks and 4.3V is equal to 881 ticks. A simple code was uploaded into the Arduino just to read the pedal sensor. To read the analogue input, a simple code was written in the structured text, which is a C++ based programming language. This can be seen in code listing 1.

Code listing 1: Arduino code for initial reading throttle value

```
int Throttle = A0;                    \\ Defines throttle analogue pin
int PedalSensor;                      \\ Variable to store throttle reading

void setup() {
  Serial.begin(9600);                 \\ Setup initialising serial monitor
}
void loop() {                         \\ Loop program
  PedalSensor = analogRead(throttle); \\ Reads input & assigns to variable
  Serial.println(PedalSensor)         \\ Prints the value to serial Monitor
}
```

This programme reads the analogue input continuously and prints the value on the serial monitor on the Arduino's IDE. Once the programme was uploaded and tested, the range of the pedal sensor was found to range from 185 ticks to 882 ticks, close to the previously calculated values. The values read on the serial monitor were found to be slightly unstable and fluctuated around ±10 ticks. To increase the reliability of the analogue read function the "analogRead" line in the programme was replaced with a function that takes an average of 10 readings instead. The code for the new read sensor function can be seen in code listing 2.

Code listing 2: Updated code for Arduino reading the throttle input

```
void ReadSensor () {
int readings[10];                          // Readings from the input
int readIndex = 0;                         // Current reading
int total = 0;                             // the running total
  total = total - readings[readIndex];     // Read from the sensor:
  readings[readIndex] = analogRead(throttle); // New value to an index
  total = total + readings[readIndex];     // Adds to the total
  readIndex = readIndex + 1;               // next position in the array
    if (readIndex >= 10) {
    readIndex = 0;                         // Resets array after
  }
```

This function creates an array and reads the analogue input 10 times before refreshing the array and starting again. Before refreshing the array, it takes an average of the readings and assigns this to the new pedal sensor variable. This code provided more stable results and the number of ticks was found to fluctuate around ±2 instead of ±10. A fluctuation of ±2 ticks results in a fluctuation of approximately 0.01V, which is an acceptable error, from the pedal sensor. The next step was to map the pedal sensor's voltage reading to a value of 0–4V in order to replicate the hand throttle input on the DC motor controller.

VII. Microcontroller Design for Input Control

To output a variable voltage from the Arduino, a digital output needs to be used with PWM capabilities in order to provide a variable DC voltage to the speed controller. When the speed controller was tested it was found that input of 0 to 4V is required to vary the motor from no speed to full speed. The Arduino has the capability of outputting 5V from any of its digital outputs and PWM can be used to vary this voltage within the range of 0 to 5V. The digital to analogue converter (DAC) in the Arduino is an 8-bit DAC which results in 256 divisions using equation 1. The Arduino has a range of 0 to 255 ticks when using the PWM output as it starts from 0 and not 1. This means that writing 255 to the digital output will provide 5V while writing 0 will provide 0V. To provide a maximum of 4V, the digital output therefore needs to be limited to 204 ticks which will output a maximum voltage of 4V. This means the input of the pedal sensor, which was found to be 185 to 882 ticks, needs to be mapped to 0 to 204 ticks when writing to the PWM output to provide a relative 0–4V, depending on the position of the pedal sensor. To achieve this, a map was used to simplify the mathematics involved and save CPU processing memory. Digital output was assigned as the throttle electric speed control (ESC) input and the pedal sensor was mapped to provide a 0 to 4V output using this pin. The new loop code for this can be seen in code listing 3.

Code listing 3: Arduino code for mapping throttle reading to the PWM pin

```
void loop() {
  ReadSensor();                        \\ Reads average pedal sensor reading
  Serial.println(PedalSensor);         \\ Prints Pedal Sensor Reading
  ThrottleOut = map(PedalSensor, 180, 880, 0, 200);  \\ Maps Pedal value
  if (ThrottleOut >0) {
  analogWrite(3, ThrottleOut);         \\ Writes PWM value to digital output
 }
}
```

This was then uploaded into the Arduino and a multimeter was attached to the PWM output. The voltage was measured while the pedal was pressed to ensure that the voltage output was within range for the ESC. As the pedal was pressed it was found that the output of the Arduino was within the 0 to 4V range and worked as expected. The PWM output of the Arduino was then connected to pin 10 and negative connections of the ESC to replace the hand throttle to test the Arduino as a throttle input. The Arduino was powered up using the PC and the DC motor hardware was powered up using the power supply. The system was tested in both forward and reverse while measuring the throttle input from the Arduino. The system ran reasonably well and up to full speed in both forward and reverse, replicating the use of the hand throttle from the previous test. Again, the speed was found to be 3100 RPM in forward and 1780 RPM in reverse using the digital tachometer with 48V being used as the supply voltage. The addition of the Arduino circuitry for the speed controller can be found in a simple schematic in figure 10. The next step is to develop a speed feedback in order to have a reference point for the PID controller.

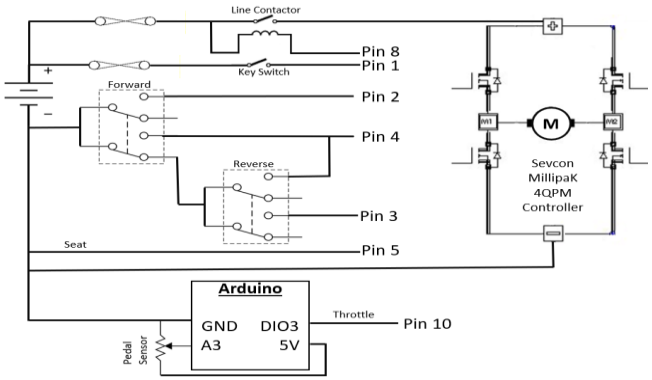

Fig. 10. Simplified schematics of system with foot pedal and Arduino.

VIII. Development of PID Control for DC Motor

The PID control developed in this system used the pedal sensor as the set point and the input taken from the speed sensor while the output is the level of PWM sent to the ESC. Instead of using the RPM of the shaft as the input of the speed sensor, miles per hour (MPH) was used in order to control the system by the final speed of the vehicle after the gearing ratio had been considered. To do this, the speed of the shaft in RPM needs to be converted to MPH by considering the diameter of the driving wheels and gearing ratio to determine the speed of the vehicle. To develop the PID control, a gearing ratio of 1:5 was used and wheels with a diameter of 18 inches as these are the intended attributes used in the mechanical build. This was fine-tuned in software when the vehicle was assembled with actual gearing ratio and wheel diameter.

The circumference of an 18 inch diameter wheel was calculated to be 1.4363 metres using the above sequence and the calculated circumference. The following formula was developed to convert the maximum RPM of the motor shaft into MPH for the vehicle.

$$MPH_{max} = 3264\,RPM \div 5\,x\,60min\,x\,1.4363metre \div 1609.3metre$$

$$MPH_{max} = 34.96\,MPH \;\therefore\; 1\,MPH = \frac{RPM}{93.36}$$

From this, it was deduced that 93.36 RPM would result in the vehicle travelling at 1 MPH. This statistic can be used in

the software to calculate the vehicle's speed at any given moment. This new MPH reading was used as the PID controller input. The pedal sensor was mapped to a maximum of 30 MPH instead of being mapping directly to the PWM output of the Arduino, and this new mapped value was used as the set point of the controller. This means that, as the pedal is pressed from nothing to fully pressed, a proportional desired MPH can be calculated and used as the set point. The Arduino's PWM output was now set as the PID controller output. This changed the control system from an open loop system to a closed loop system able to try to attain the speed set by the pedal sensor via altering the PWM output accordingly. This was implemented into the Arduino coding with the proportional constant (Kp) set to 1 and the integral and derivative constants (Ki and Kd) set to 0 ready for the Ziegler Nicholas tuning method to be used. This code was uploaded into the Arduino and tested with only the Kp set to 1. To test the system, the pedal sensor was removed through software and the set point was set to 15 MPH for stability.

Figure 11 shows the plot of the set point and actual motor speed for the various PID constants and figure 12 shows the plot of actual speed versus variable set point.

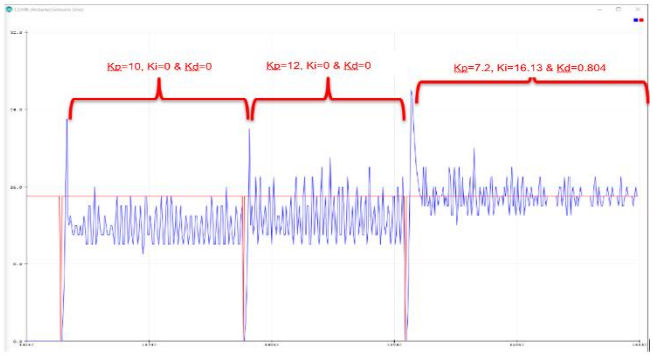

Fig. 11. Actual speed vs setpoint from Arduino IDE (Kp=7.3, Ki=16.13 & Kd=0.804).

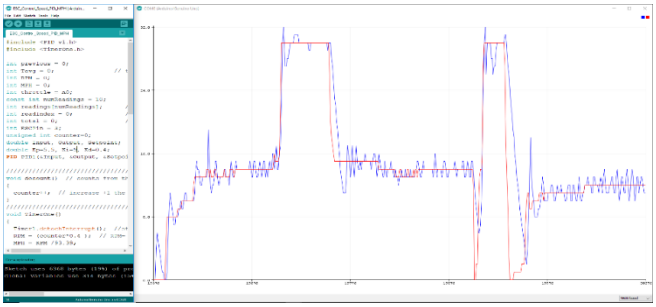

Fig. 12. Actual speed vs variable setpoint (Kp=5.5, Ki=9 & Kd=0.4).

IX. DEVELOMNET OF USER INTERFACE

A user interface was developed to show the driver of the vehicle key variables such as the RPM of the motor, the speed of the vehicle in MPH and the current being drawn and voltage applied to the motor alongside overall power. For this, an Arduino 3.5-inch TFT touch screen was used. This was a comprehensive colour screen to develop a user interface for the vehicle. The screen was compatible with the Arduino Nano, Uno and Mega and was developed separately on an Arduino Nano to begin with. It came with a CD with many examples and tutorials to follow and was used to learn how to use and operate it. The screen can be seen in figure 13.

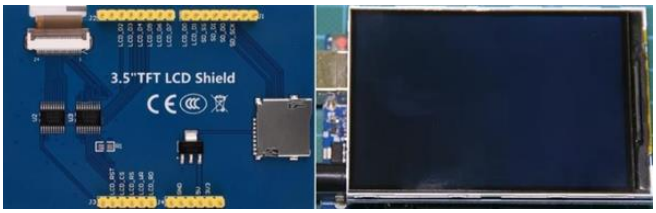

Fig. 13. Arduino 3.5" TFT touch screen used for the user interface (Back and Front).

The screen was connected to the Arduino Nano following the pinout diagrams seen in figure 14. The examples on the CD were uploaded into the Arduino and each one was analysed separately to understand how to manipulate the code for the needs of this research. A user interface was developed that showed two different screens that could be switched with a momentary button. The first screen showed the speed of the vehicle in MPH on a dial and the RPM on a vertical bar graph. The second screen showed the power and current drawn by the motor as well as the battery voltage. These are shown in figure 14.

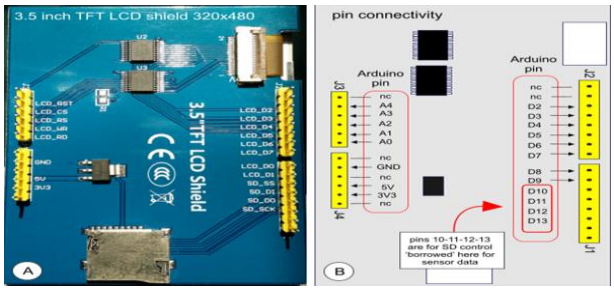

Fig. 14. Arduino screen pinout for Arduino Mega.

A user interface was developed to show the driver of the vehicle key variables such as the RPM of the motor, the speed of the vehicle in MPH along with the current being drawn and voltage being applied to the motor along with overall power. For this an Arduino 3.5-inch TFT Touch Screen was used. This was a comprehensive colour screen that was used to develop a user interface for the vehicle. The screen was compatible with the Arduino Nano, Uno and Mega and was developed separately on an Arduino Nano to begin with. The screen came with a CD with many examples and tutorials to follow and was used to learn how to use and operate the screen. The screen can be seen in figure 15.

Fig. 15. User interface developed for speed, battery voltage and current.

X. BLUETOOTH SYSTEM FOR USER INTERFACE COMMUNICATION

A Bluetooth system was developed to send data between the user interface and the main microcontroller controlling the ESC. Figure 16 shows a simple diagram of the Bluetooth system developed using two Bluetooth modules (HC05 and HC06).

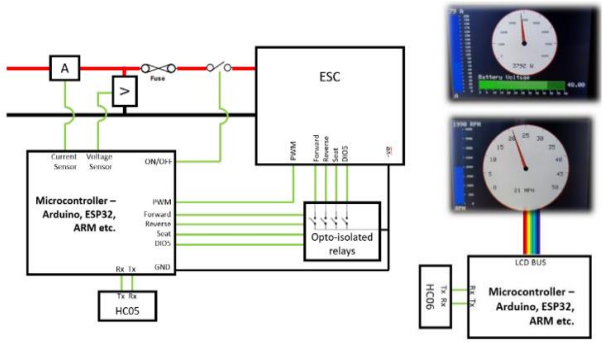

Fig. 16. Schematics of Bluetooth connected LCD screen & opto-isolated relays for digital IO.

XI. Implementation and Testig

An off-road go-kart frame is used. The frame has been slightly modified to date to now include a live rear axle and back and front suspension. This frame is big enough to support one person and weighs 50kg before any development on the frame. Figure 17 shows before and after photos of the work done so far on the go-kart frame.

Fig. 17. Prototype before and after modification.

The first stage of the rig development was to take the motor and install it onto a sturdy right-angle frame. This was attached to a base plate and the rest of the electronics were attached to this. The hall sensor and magnetic disk were attached to the shaft of the motor, ensuring that the hall sensor and magnets were within a reasonable distance and the south side of the magnets faced the hall sensor. The speed controller was added to the rig with an aluminium block used as a temporary heatsink. The aluminium block was cut down to size with four holes drilled in and tapped to M6 to enable tightening down of the electronic speed controller to it. Before this was done, a layer of heatsink paste was applied to the heatsink to help the thermal flow of any excess heat. The ESC was then tightened to the aluminium bleak on the rig with a torque of 6 nm, as suggested in the manual. The digital IO of the ESC was connected near to the ESC and all wires made as short as possible. The direction switches were installed and the ignition switch was changed to a key switch to be incorporated into the final vehicle. A DC to DC convertor was incorporated into the system to provide the Arduino with 12V from the battery system. The Arduino Mega was attached to the base of the rig and all the connections for the ESC, foot pedal, hall sensor and current sensors were made. The screen was attached to a wooden base and attached to the rig. This was wired to the Arduino and all the wiring was checked to ensure that it was all connected correctly and there was no risk of damaging any of the equipment on the first power-up. Four lead acid batteries were placed on the bottom shelf of the trolley and connected to the ESC. An opto-electronic controlled relay circuit was sourced which could handle the voltage and current of an electric start circuit of a generator or a petrol engine depending on the build. This allowed for full control of power to the electric start circuitry of the IC engine while being electrically isolated from the microcontroller in any cases of high potential difference. This was tested with a small DC motor, replicating the starter motor – to activate the relays a high digital output is needed from the microcontroller. These worked extremely well and will be used to control any digital signals needed to control the ESC such as forward, reverse, start and reset.

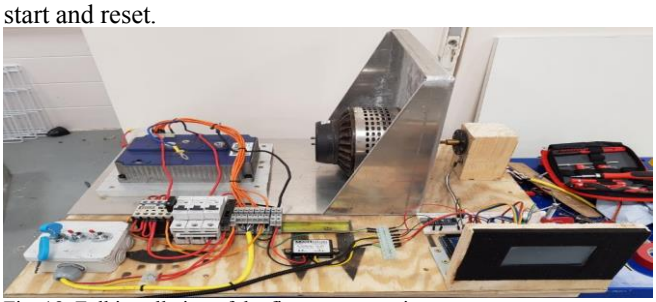

Fig. 18. Full installation of the first prototype rig.

XII. Conclusion

This paper presented the design and experimental verification of the DC electric motor to control the speed for a HEV. The simulation results show that PID control algorithm can improve the performance of DC electric motor speed at varied set points, Kp, Ki, Kd, to obtain a reliable and stable speed. It also clearly shows that the user interface provides better communication between the Arduino and PID controller parameters for maximum and stable speed and operational safety. The implementation and testing outcomes proved that the proposed prototype and controller are efficient and accurate for HEVs.

References

[1] F. Badin, J. Scordia, and R. Trigui, “Hybrid electric vehicles energy consumption decrease according to drive train architecture, energy management and vehicle use,” IET Hybrid Veh. Conf., pp. 213-223, 2006.

[2] C. C. Chan, and K. T. Chau, “An overview of power electronics in electric vehicles,” IEEE Trans. on Industrial Electronics, vol. 44, issue: 1, 1997.

[3] Z. Q. Zhu, and D. Howe, “Electrical machines and drives for electric, hybrid, and fuel cell vehicles,” IEEE Proceedings, vol. 95, issue: 4, 2007.

[4] P. Pillay, and R. Krishnan, “Modeling, simulation, and analysis of permanent-magnet motor drives, Part I: The permanent-magnet synchronous motor drive, IEEE Trans. on Industry Applications, vol. 25, no. 2, pp. 265-273, 1989.

[5] P. Pillay, and R. Krishnan, “Modeling simulation, and analysis of permanent-magnet motor drives, Part 11: The brushless DC motor drive, IEEE Trans. on Industry Applications, vol. 25, no. 2, pp. 274-279, 1989.

[6] R. Shanmugasundram, M. Zakariah, and N. Yadainah, “Implementation and performance analysis of digital controllers for brushless DC motor drives, IEEE/ASME Trans. on Mechatronics, vol. 19, issue: 1, 2014.

[7] A. K. Yadav, P. Gaur, S. K. Jha, J. R. P. Gupta, and A.P. Mittal, “Optimal speed control of hybrid electric vehicles, Journal of Power Electronics, vol. 11, no. 4, pp. 393-400, 2011.

[8] A. K. Yadav, and P. Gaur., “An optimized and improved STF-PID speed control of throttle controlled HEV, The Arabian Journal for Science and Engineering, vol. 41, no. 5, pp. 3749-3760, 2016.

[9] K. C. Prajapati, R. Patel, and R. Sagar, “Hybrid vehicle: A study on technology” International Journal of Engineering Research & Technology, vol. 3, issue: 12, pp. 1076-1082, 2014.